AMC CSS Achieves CMM Level 3

The Air Mobility Command (AMC) Computer Systems Squadron (CSS), Scott Air Force Base, Ill. received a Level 3 rating during a Software Engineering Institute (SEI) Capability Maturity Model (CMM) assessment. The CSS currently has over 450 employees dedicated to developing, maintaining, and enhancing transportation and command and control software systems for AMC. The assessment culminated 17 months of dedicated hard work.

One requirement to achieve Level 3 was to develop and maintain a usable set of software process assets that improve performance across all projects and provide a basis for cumulative, long-term organizational benefit. They developed a process asset library (PAL) located on the Web at http://cpssweb.safb.af.mil:81/pal/pal_home.htm. The AMC CSS PAL is accessible to everyone within the military and government Internet domains.

Martino provided valuable insight that is reflected in some form in this article. ◆

About the Author

Gary A. Ham is a senior research scientist for Battelle Memorial Institute, National Security Division, Information Systems Engineering and Process Modernization Department in Arlington, Va. A former Marine Corps comptroller and Naval Academy computer science instructor, he currently researches value metrics definition processes to support object-oriented requirements analysis and design of DoD systems. He has a bachelor's degree in economics from Whitman College in Walla Walla, Wash. and a master's degree (with distinction) in information systems management from the Naval Postgraduate School in Monterey, Calif. He is currently a doctoral candidate in information technology at George Mason University in Fairfax, Va.

Principal Research Scientist
Battelle Memorial Institute
2101 Wilson Blvd., Suite 800
Arlington, VA 22201-3008
Voice: 703-575-2118
Fax: 703-671-9180
E-mail: ham@battelle.org

References

1. Research Report, "Chaos," The Standish Group, 1995, http://www.standishgroup.com/chaos.html.
2. Jacobson, Ivar, Martin Griss, and Patrick Jonsson, *Software Reuse: Architecture, Process, and Organization for Business Success*, ACM Press, New York, N.Y., 1997.
3. Fenton, Norman E., *Software Metrics, a Rigorous Approach*, Chapman and Hall, London, 1994.

Notes

1. Ivar Jacobson's basic definition differs slightly: "A use case is a sequence of transactions performed by a system, which yields an observable result of value for a particular actor" [2]. For our purposes, an actor is defined as a participant in a use case event, as an instigator, a provider of service or product, or as a recipient of that service or product.
2. If this sounds a little like Integration Definition for Function Modeling (IDEF0), it should. IDEF0, with a difference in focus from functional decomposition to product or service identification, can effectively be used to identify mission-focused use cases. The required change in mindset may be difficult for traditional IDEF0 modelers. It was for me. If you can make the transition, however, a whole new approach to software metrics based on activity-based costing becomes available.
3. The particular form that a use case should take is less important than the content. The only requirement is a consistent presentation of use case contents that provides clear understandability by subject matter experts. The use of formal notation languages, e.g., Unified Modeling Language and predicate logic, should be left out unless the user community is fully conversant in the notation presented. We use a standard format for our use cases. This "standard" has, however, been adjusted in each analysis iteration to better meet the understandability needs of our validating users.
4. Business rules are defined to be requirements that contain a conditional phrase, e.g., "if," or "then." Business rules are designed to govern the actions of an event or events, either singly or grouped in a rule base. In some references, nonconditional rules are referred to as business rules. My current project merely calls such rules requirements. We feel the distinction is important because business rule sets can be used within rule engines to process events depending on condition. Straight requirements act regardless of condition.
5. Although analogous, this is not the same as the class, responsibility, and collaboration approach. We are defining roles that will probably (but may not) be assigned to classes as part of the design process. We do not try for class definition in analysis use case development. We save that for design, when architectural dependency issues are more completely specified.
6. Yet, we have used this approach extensively for project management. All of our task statement development and project work breakdown structures are based on the definition of responsibilities and collaborations between project teams where project teams are recognized as actors or classes in the object-oriented sense. Project management is object management to the extent that Gantt charts are defined by a composition of sequence diagrams developed from the original collaboration diagrams.